

E-Privacy – Privacy in the Electronic Society

Encryption, Verifiable Encryption,
and Anonymous Communication

Jan Camenisch

IBM Zurich Research Laboratory

Rüschlikon

jca@zurich.ibm.com

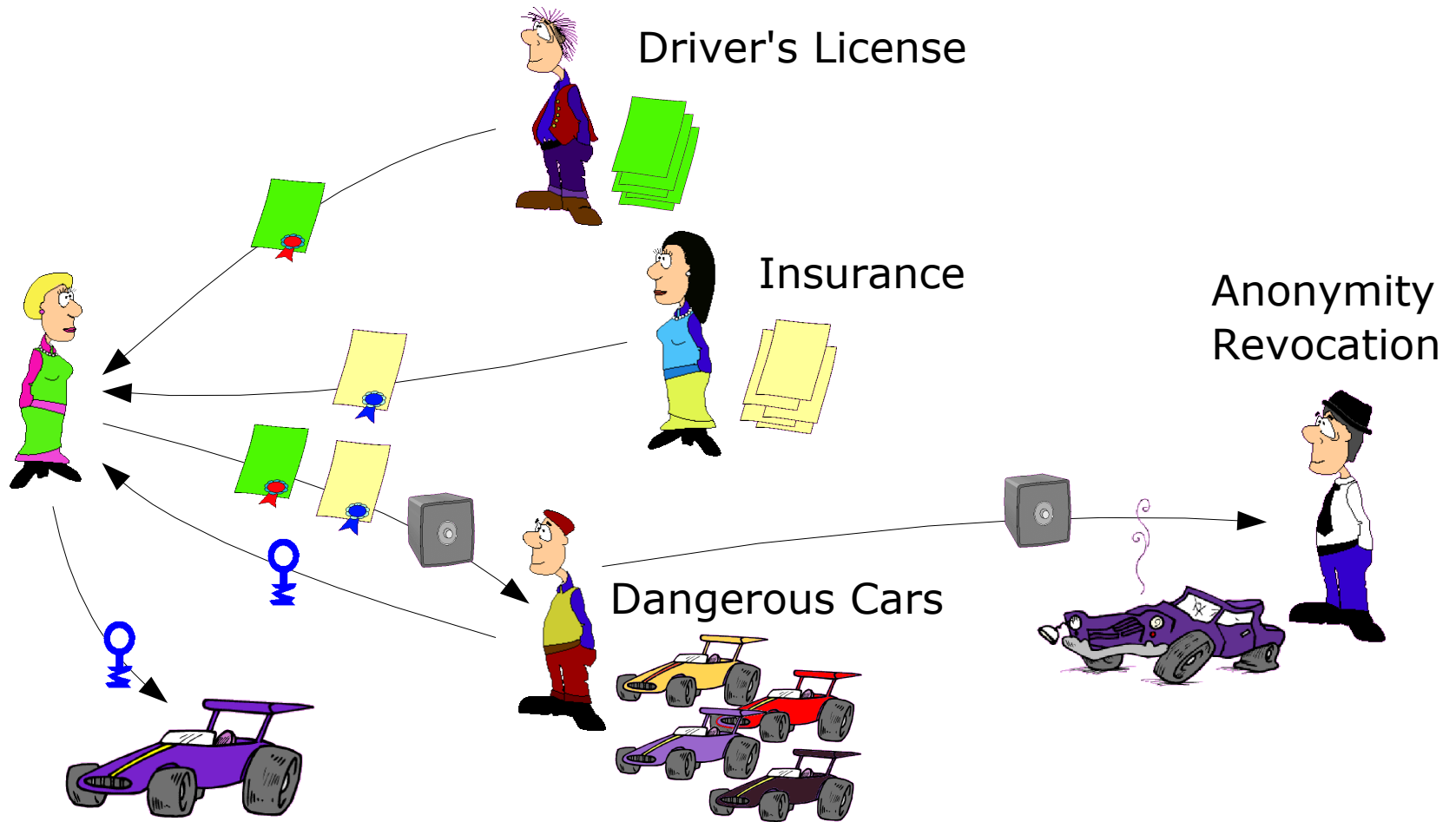
Outline

- Credential Systems & What's still needed 4 them
- Public Key Encryption: Definition and Schemes
- Verifiable Encryption
- Anonymous Communication
 - Provable Mixnets
 - Onion Routing
 - Dining Cryptographers
 - Crowds

Recap Technologies

Proof Protocols, Commitments, and Signatures.

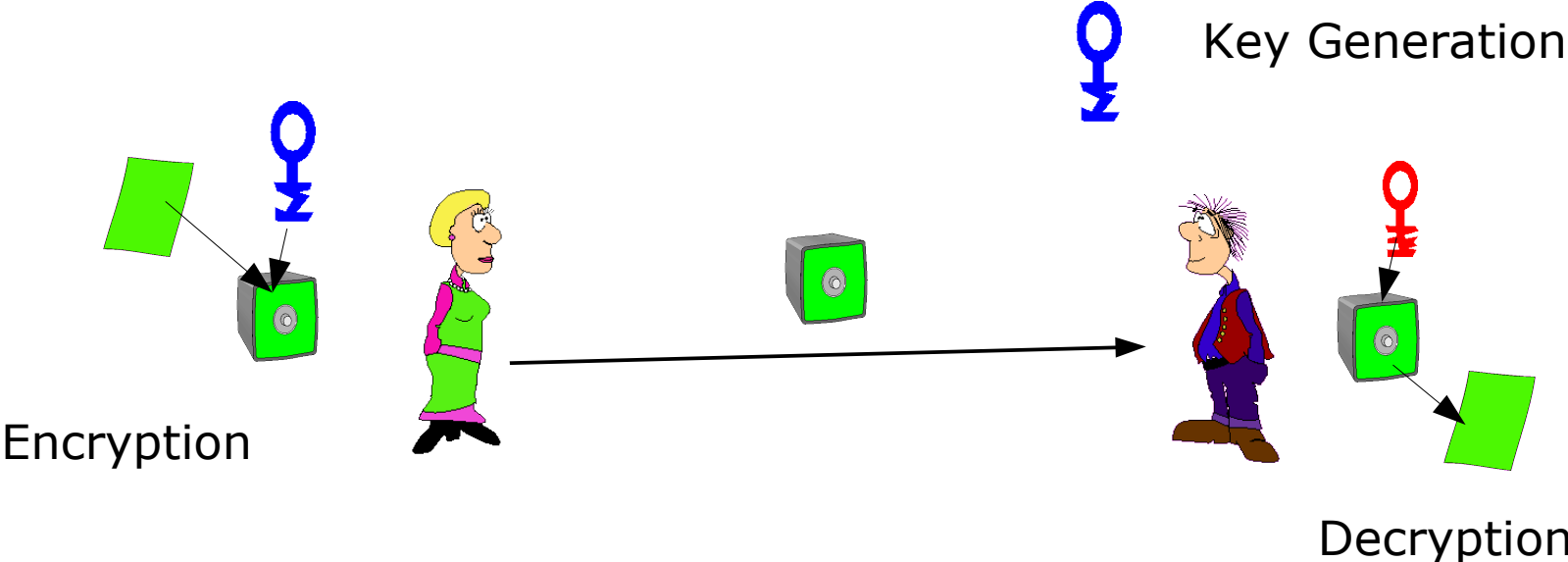
Example cont'



Public Key Encryption

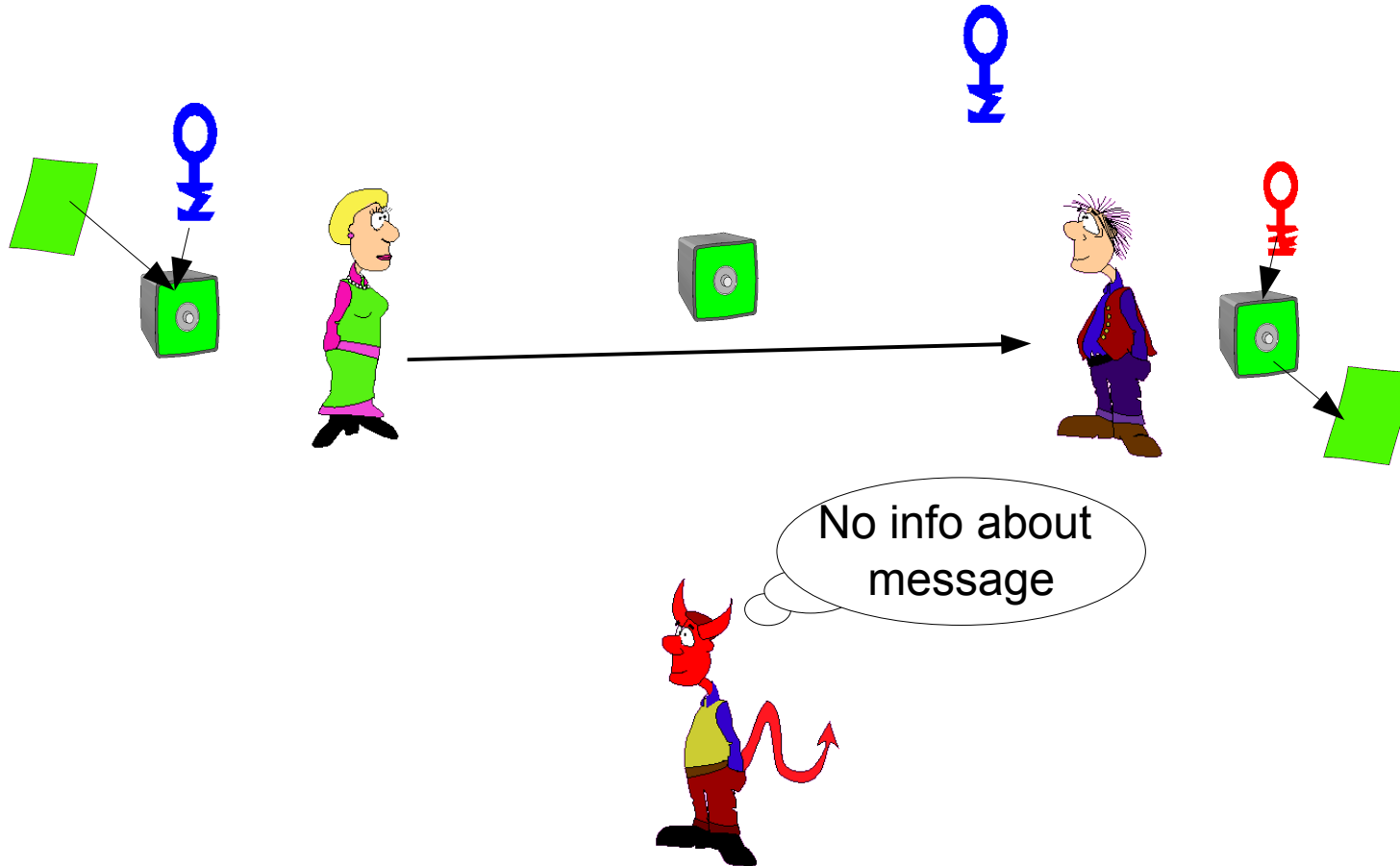
Definition and Schemes

Public Key Encryption



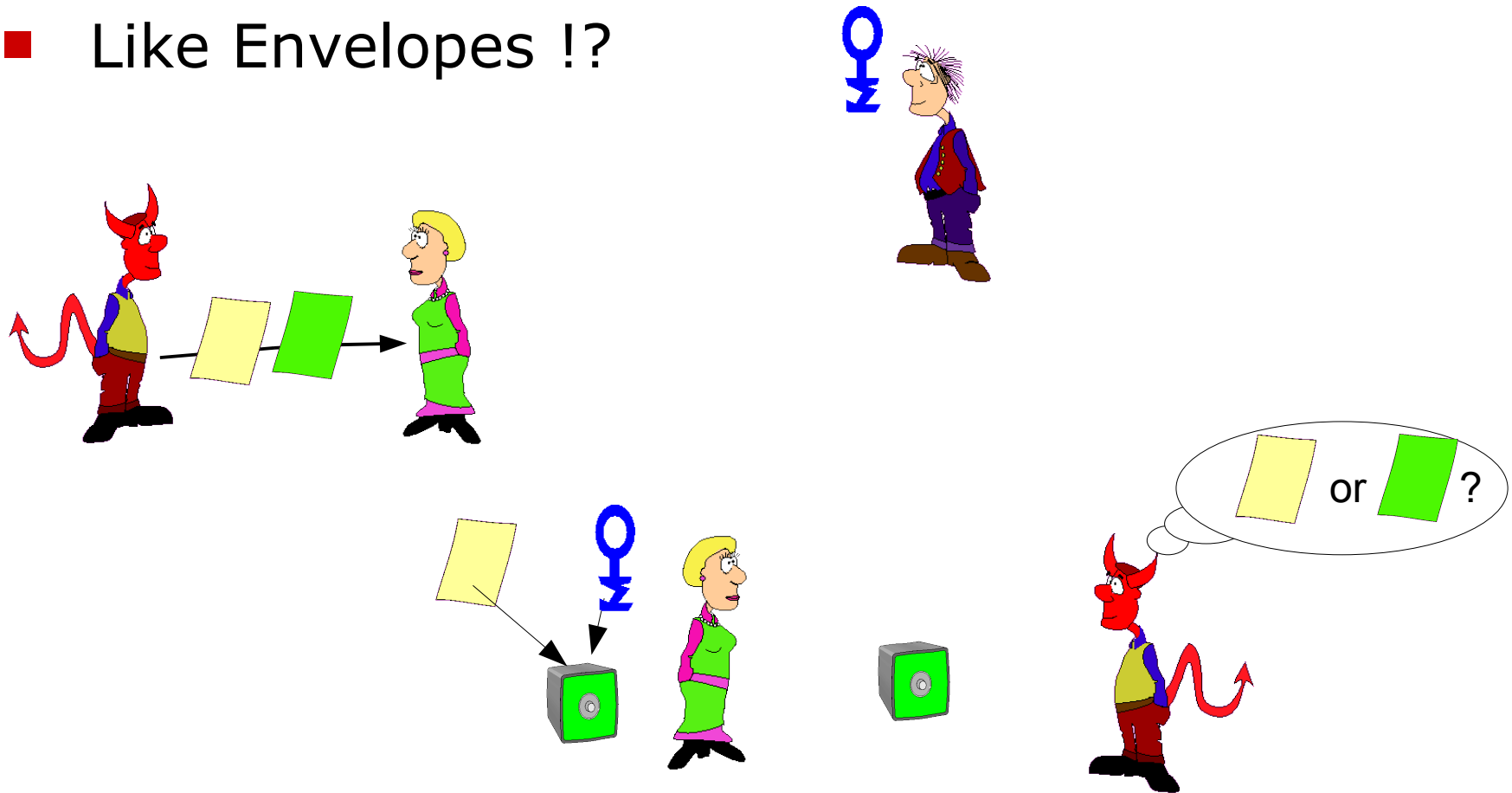
Security

- Like Envelopes !?



Security

- Like Envelopes !?



- Semantic Security (secure if used once only.)

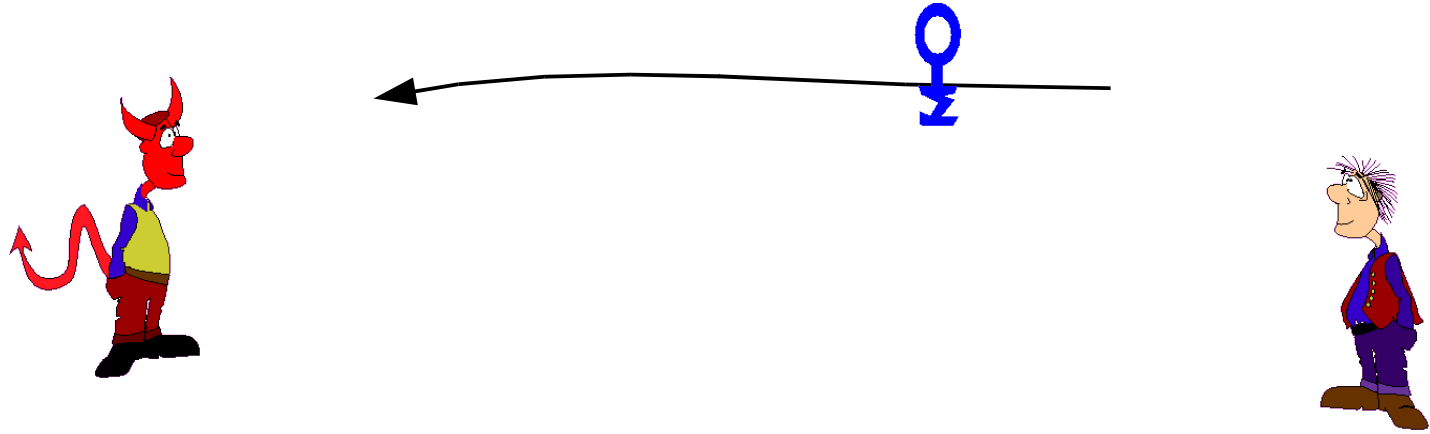
ElGamal Encryption

- Group $G = \langle g \rangle$ of order q
- Secret Key Group $x \in \{1, \dots, q\}$; Public key $y = g^x$
- To encrypt message $m \in \langle g \rangle$:
 - choose random $r \in \{1, \dots, q\}$;
 - compute $c = (y^r m, g^r)$
- To decrypt ciphertext $c = (c_1, c_2)$
 - set $m = c_1 c_2^{-x} = y^r m g^{-xr} = y^{r-r} m = m$

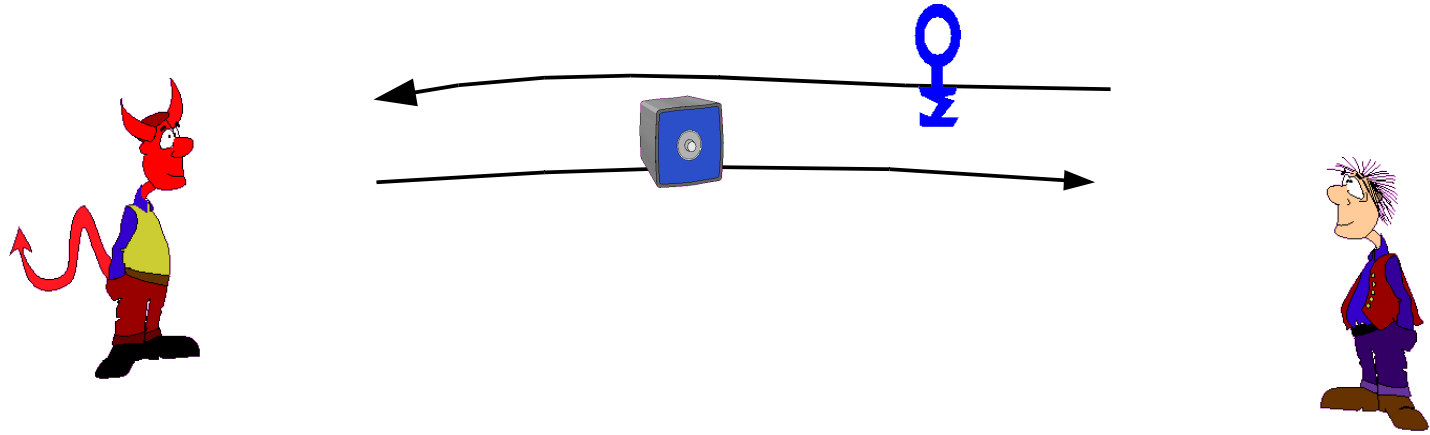
ElGamal Encryption: Security

- Semantic Security:
does ciphertext $c = (c_1, c_2)$ contain m_1 or m_2 ?
- Consider $c = (y^r m, g^r) = (c_1, c_2)$:
 c encrypts m_i iff $\log_y c_1/m_i = \log_g c_2$.
I.e., if $\log_y c_1/m_i = \log_y y^r m/m_i = \log_g g^r = \log_g c_2$.
- Deciding if $\log_y c_1/m_i = \log_g c_2$ holds for random y, g, c_1, c_2 is assumed to be hard.
- Indeed is equal to the so-called the *Decisional Diffie-Hellman Assumption*, stating it's hard given a, b, c, d to decide whether $\log_a b = \log_c d$.

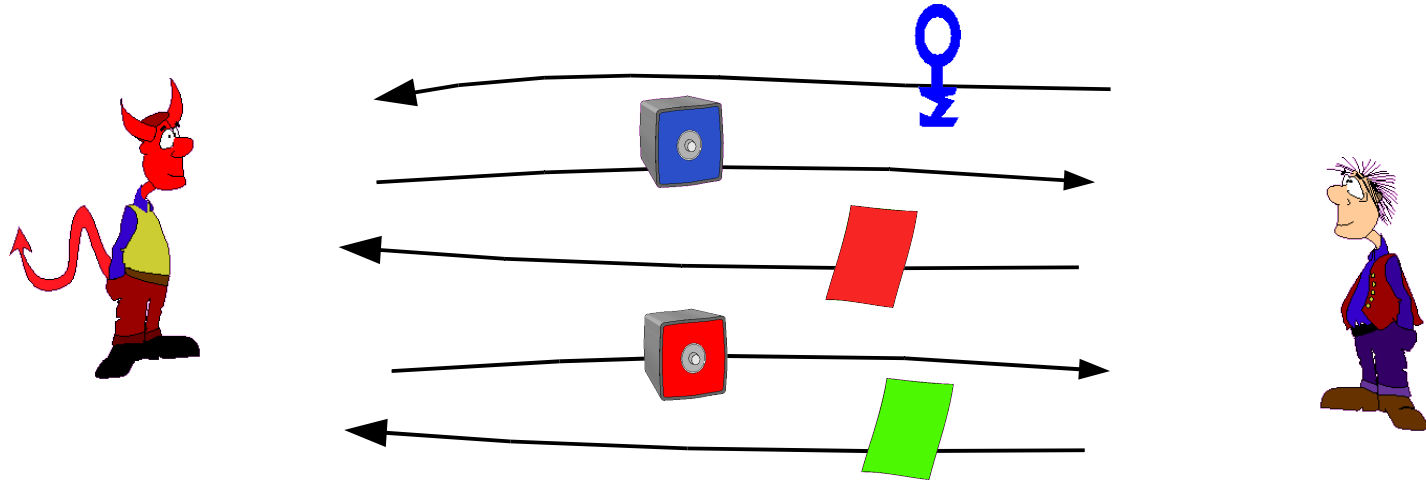
Security: Use Many Times



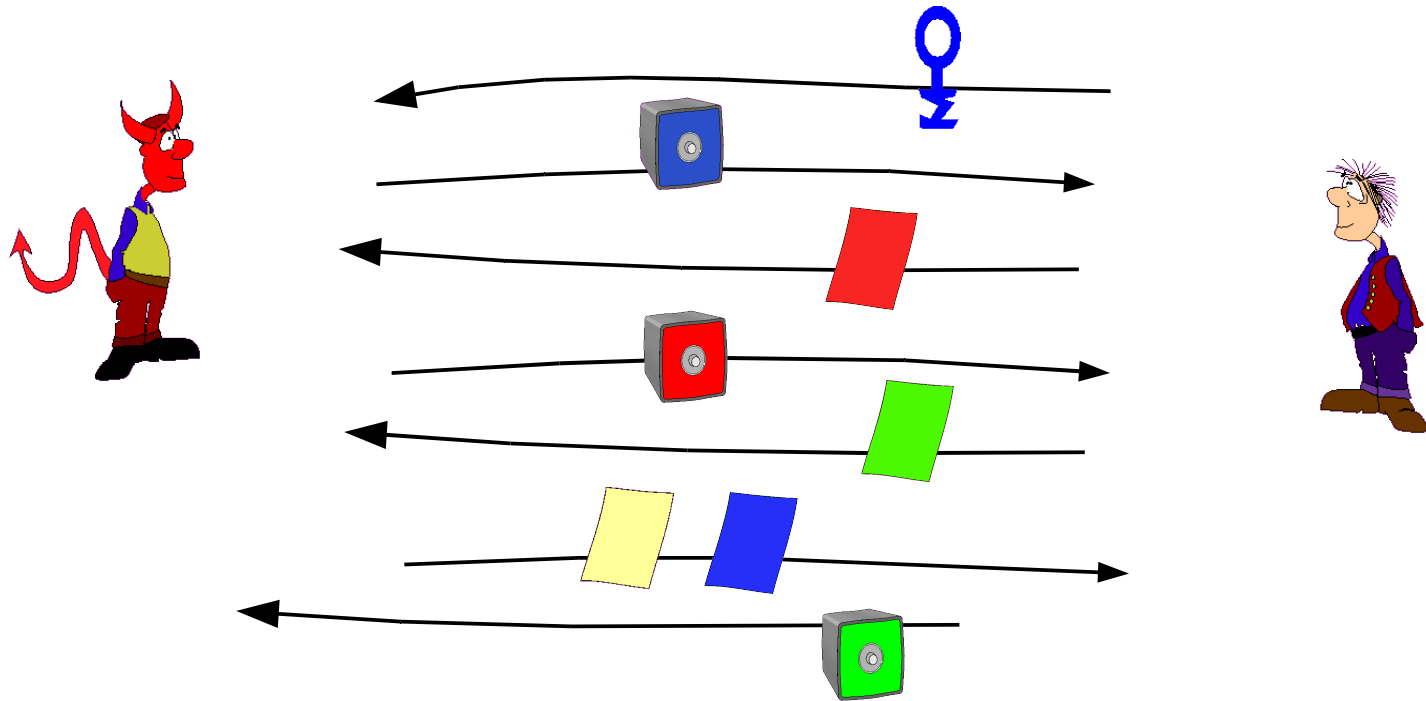
Security: Use Many Times



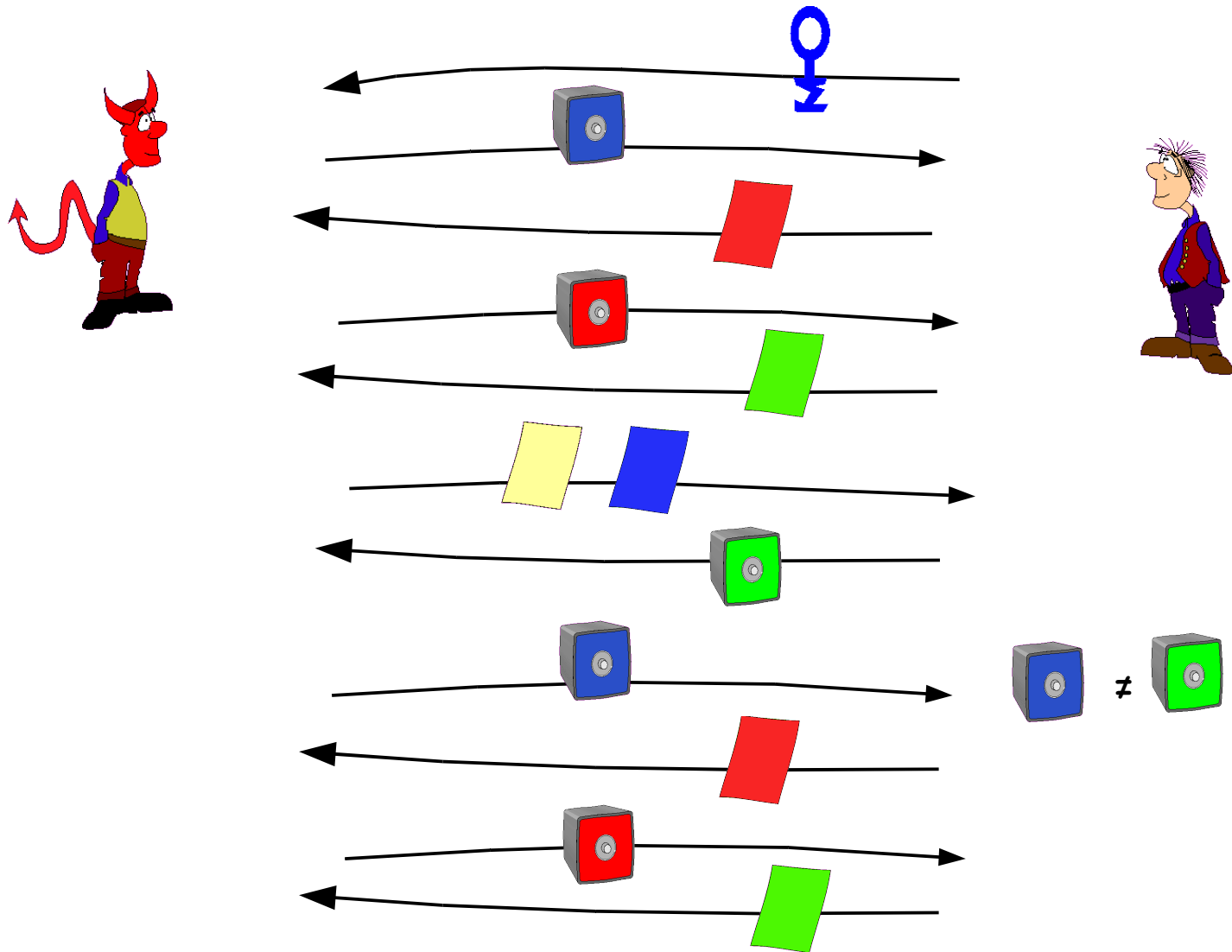
Security: Use Many Times



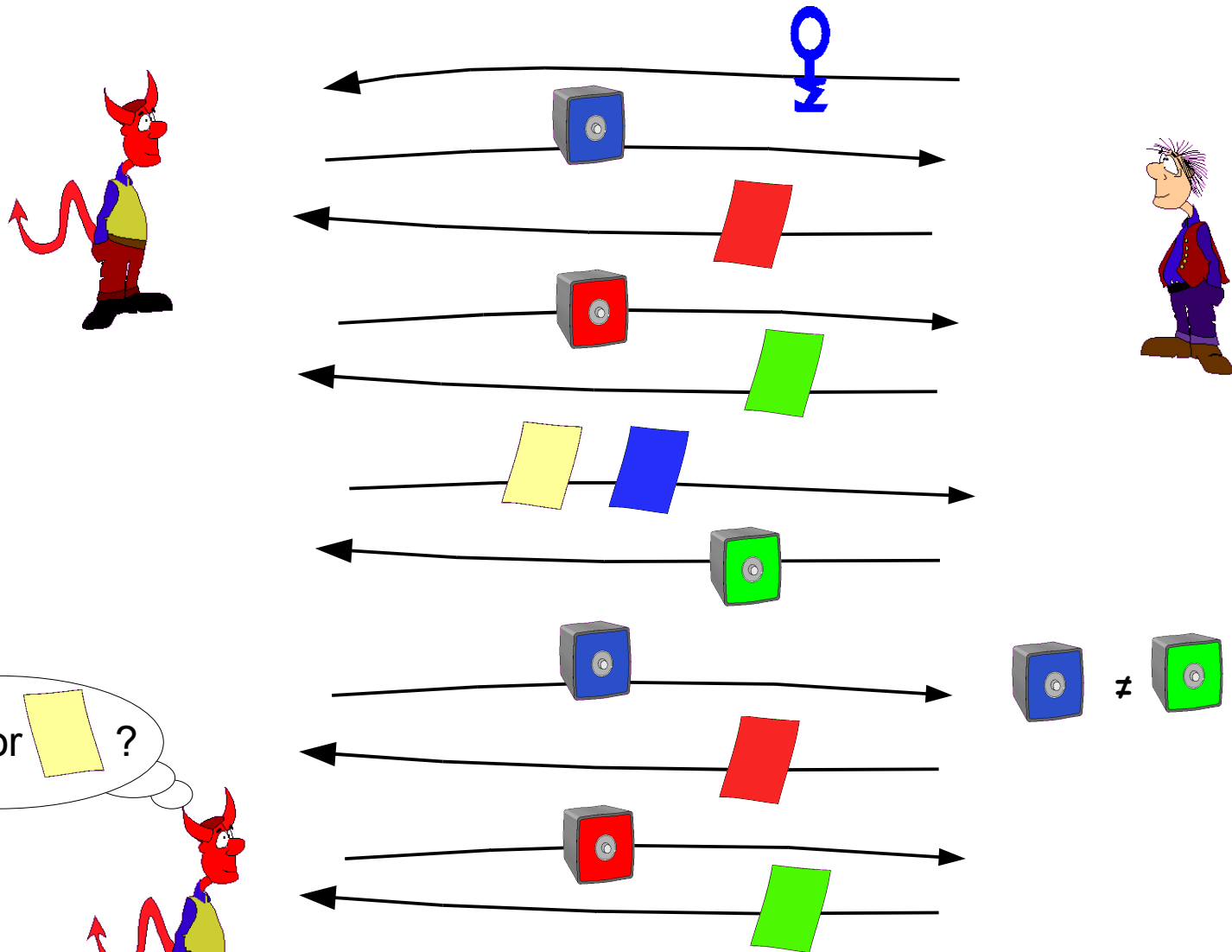
Security: Use Many Times



Security: Use Many Times



Security: Use Many Times

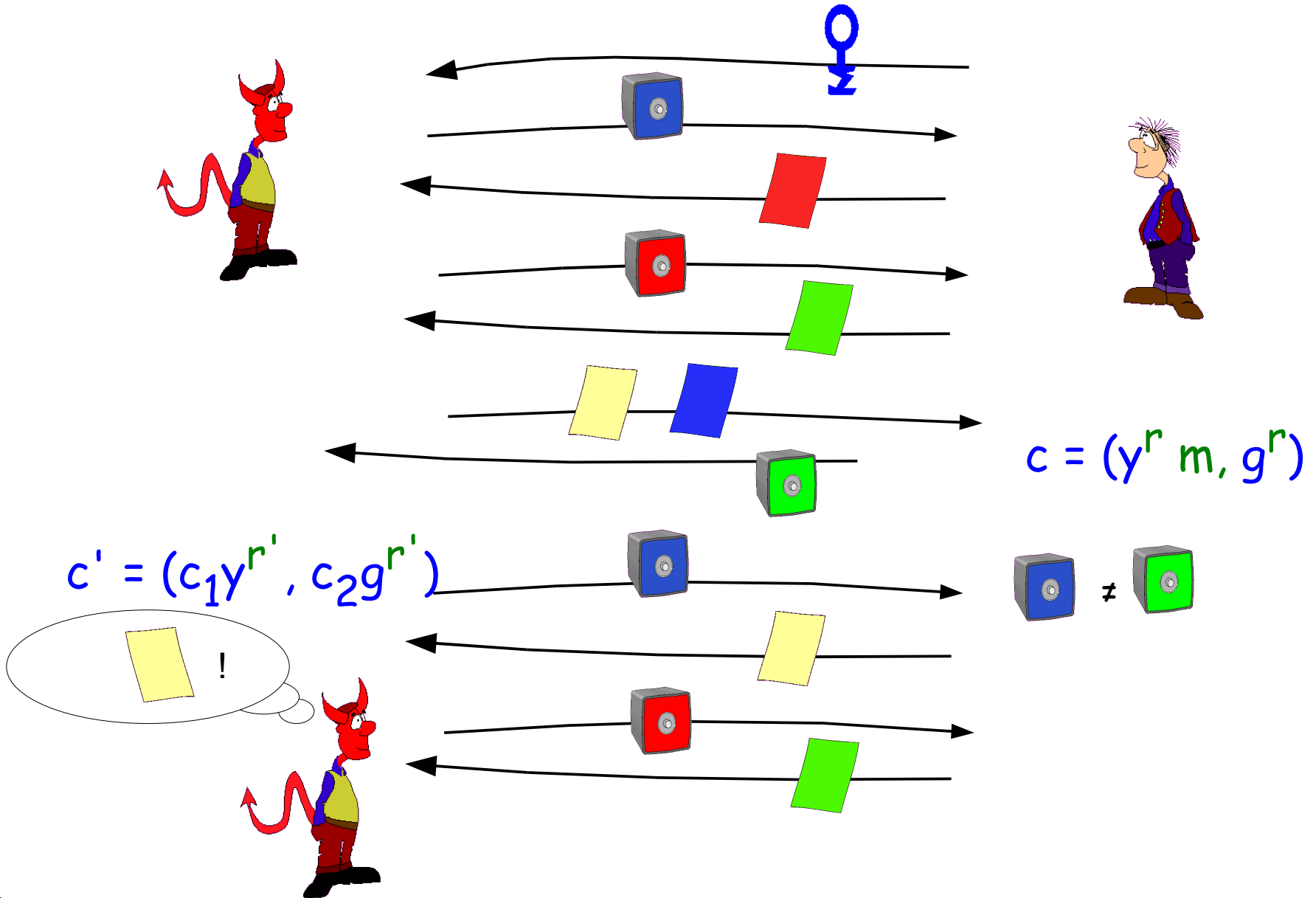


Chosen ciphertext attacks.
Strongest possible definition!

ElGamal Encryption: Security

- Recall: ElGamal encryption of m is $c = (y^r m, g^r)$
- ElGamal is randomizable:
 - given $c = (y^r m, g^r) = (c_1, c_2)$
 - compute $c' = (c_1 y^{r'}, c_2 g^{r'}) (= (y^{r+r'} m, g^{r+r'}))$
 - Ciphertext c' is indistinguishable from c
 - (So ElGamal is malleable)
- Can be good in some cases (one can produce a random version of the same ciphertext that cannot be linked to the original one).
- But also means it is not secure against so-called chosen ciphertext attacks (see next slide)!

ElGamal is not secure against CCA



Modified ElGamal Encryption

- Group $G = \langle g \rangle$ of order q
- Secret Key Group $x_1, x_2 \in \{1, \dots, q\}$;
Public key $y_1 = g^{x_1}, y_2 = g^{x_2}$
- To encrypt message $m \in \langle g \rangle$:
 - choose random $r_1, r_2 \in \{1, \dots, q\}$;
 - compute $c = (y_1^{r_1} m, g^{r_1}, y_2^{r_2} m, g^{r_2}, p) = (c_1, \dots, c_5)$
where

$$p = \text{PK}\{(\alpha, \beta): c_1/c_3 = y_1^\alpha / y_2^\beta \wedge c_2 = g^\alpha \wedge c_4 = g^\beta\} (c_1, \dots, c_5, L)$$

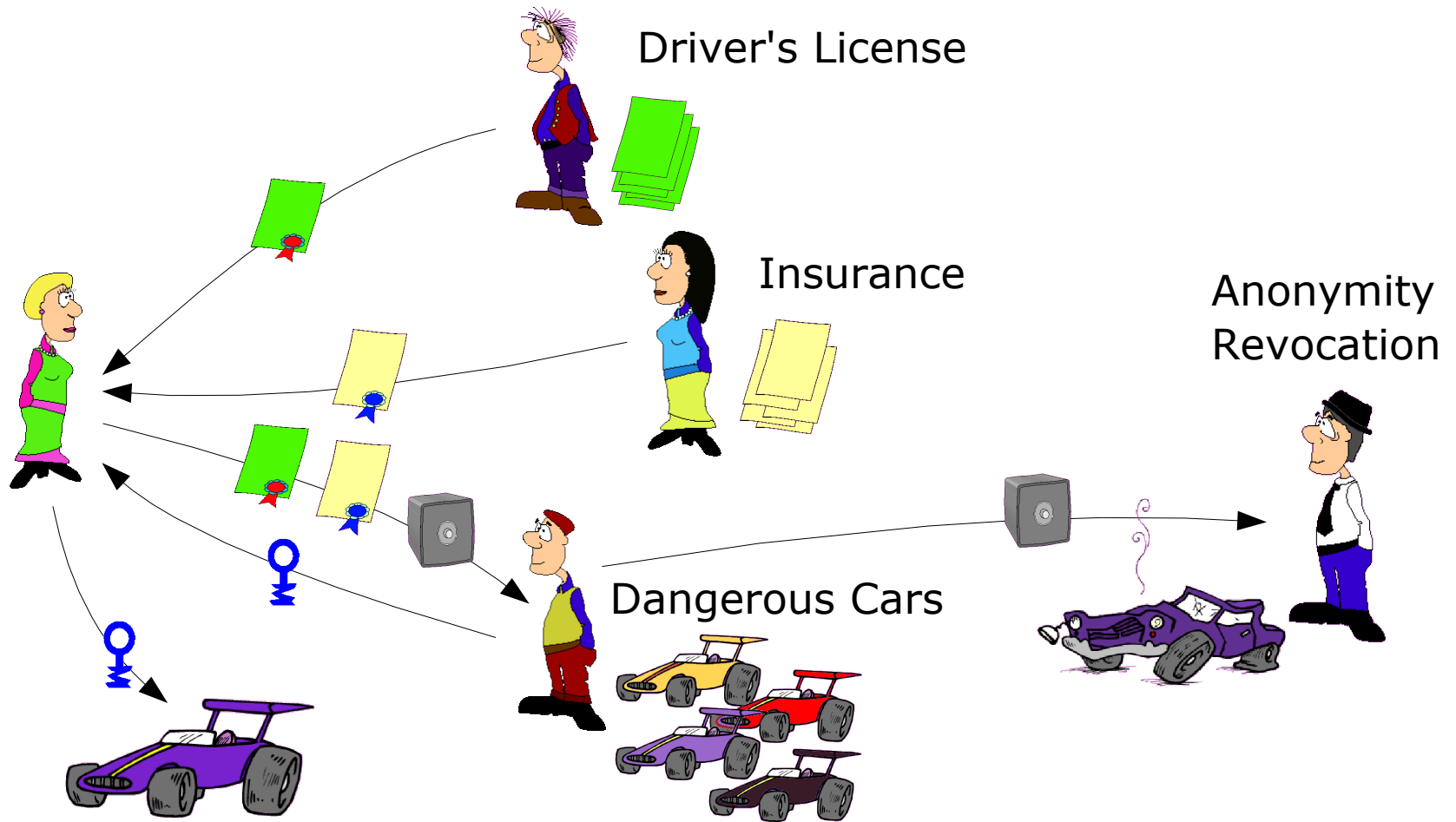
and L is a *Label* that can be attached to encryption (i.e., we use non-interactive Fiat-Shamir heuristic /signature version of proof).

- To decrypt: verify proof and decrypt parts
- Security: proof p "requires" knowledge of r_1, r_2

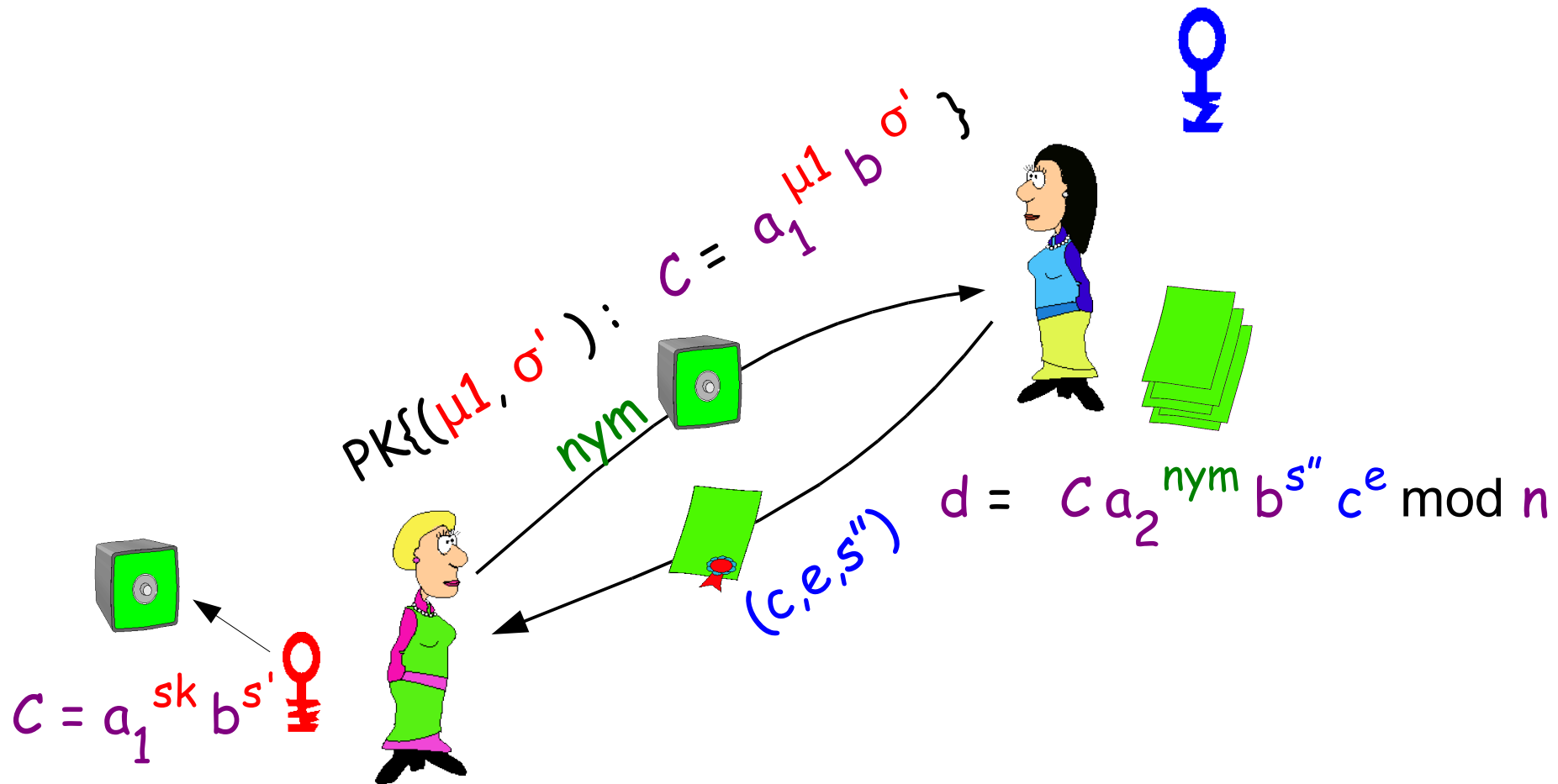
Technologies

Verifiable Encryption

Example cont'

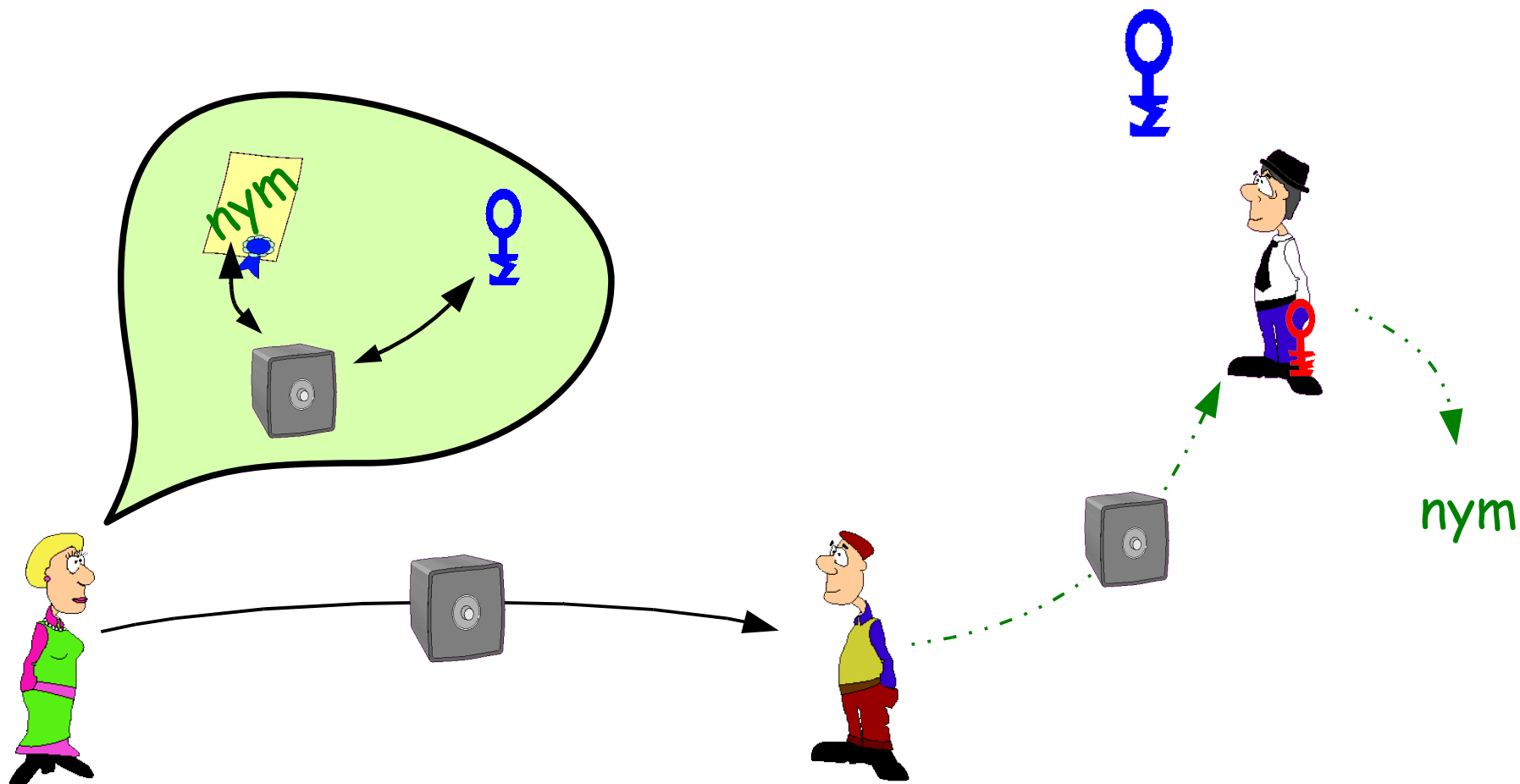


Verifiably Encrypt Signed Messages....



$$d = c^e a_1^{sk} a_2^{nym} b^{s'' + s'} \pmod n$$

Verifiable Encryption



Technologies

Verifiable Encryption of Group Elements

Verifiable ElGamal Encryption

- Group $G = \langle g \rangle$ of order q ; Public key $\gamma = g^x$
- Encrypt g^{nym} instead of nym
- To encrypt message
 - choose random $r \in \{1, \dots, q\}$;
 - compute $c = (\gamma^r g^{nym}, g^r) = (c_1, c_2)$
 - PK $\{(\varepsilon, \mu, \delta, \rho, \sigma) :$
$$d = c_1^\varepsilon a_1^\delta a_2^\mu b^\sigma \pmod{n} \wedge$$
$$c_1 = \gamma^\rho g^\mu \wedge c_2 = g^\rho\}$$
- Similarly for the 2-keys & proof version of ElGamal

Technologies

Verifiable Encryption of Discrete
Logarithms

The Decision Composite Residuosity Assumpt.

The DCR Problem: *Given n and x , decide whether or not*

$$x \in (Z_{n^2}^*)^n .$$

- Introduced by Paillier '99.
- If $n = (2p'+1)(2q'+1)$ then $Z_{n^2}^* = Z_2 \times Z_2 \times Z_n \times Z_{p'q'}$.
- $(1+n)^u = (1+un) \pmod{n^2}$.

An Encryption Scheme

Public Key: n and $g, Y_1, Y_2, Y_3 \in \langle (g')^{2n} \rangle$, where $g' \in Z_{n^2}^*$,

Secret Key: $x_i = \log Y_i$

Encryption message $m \in [0, n]$ under label L :

- choose $r \in [0, n/4]$
- $u := g^r$, $e := Y_1^r (1+n)^m$, $v := \text{abs}(Y_2 Y_3^{H(u,e,L)})^r$
- output (u, e, v) .

where $\text{abs}()$ maps $(a \bmod n^2)$ to $(n^2 - a \bmod n^2)$ if $a > n^2/2$, and $(a \bmod n^2)$ otherwise, where $0 < a < n^2$.

An Encryption Scheme

Decryption of ciphertext (u, e, v) under label L :

- verify $v = \text{abs}(v)$ and $u^{2(x_2 + H(u, e, L)x_3)} = v^2$.
- $\hat{c} := (e/u^{x_1})^{2^t}$ where $t = 2^{-1} \bmod n$,
- if $n \mid (\hat{c}-1)$ output $m := (\hat{c}-1)/n$, otherwise output \perp

Theorem: *Encryption scheme is secure against adaptively chosen ciphertext attacks under DCR assumption.*

Verifiable Encryption of a Discrete Logarithm

Let $\hat{c} = \hat{g}^m \hat{h}^s$ be a commitment to m and (u, v, e) be an encryption of m .

Prover executes

$$\text{PK}\{(\rho, \mu, \sigma, \tau) : \hat{c} = \hat{g}^\mu \hat{h}^\sigma \wedge$$

$$u^2 = g^{2\rho} \wedge e^2 = y_1^{2\rho} (1+n)^{2\mu} \wedge v^2 = (y_2 y_3^{H(u,e,L)})^{2\rho} \}$$

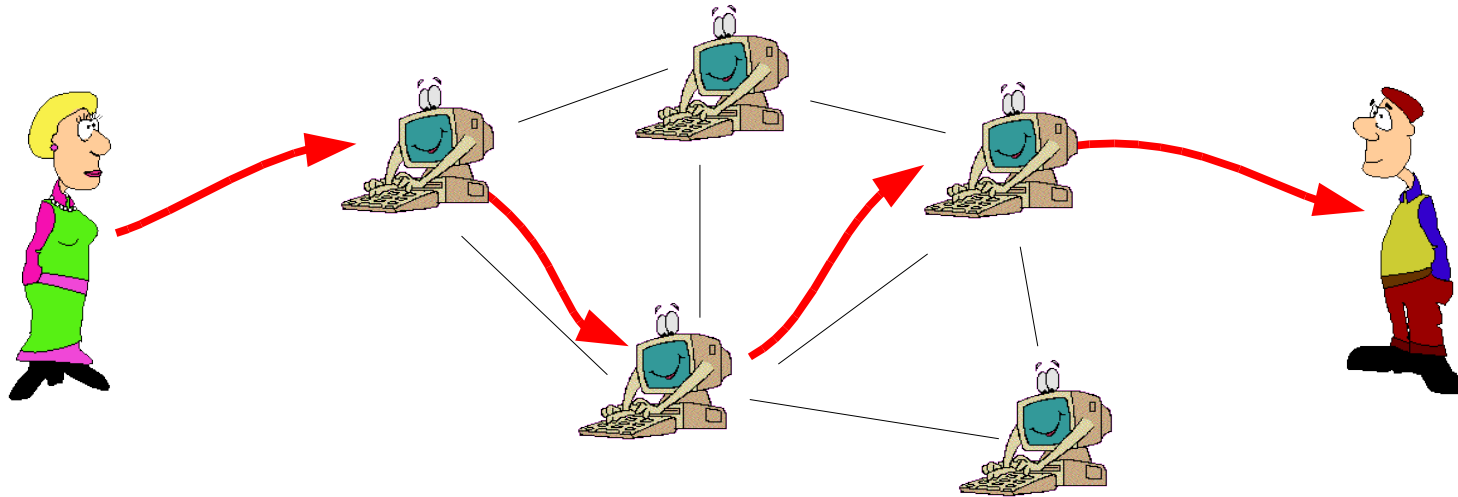
Technologies

Anonymous Communication
Mixnet Works

Anonymous Communication: Outline

- The Problem
- Mix-networks
- Provable Mix-networks
- Onion routing
- Alternatives
 - Dining Cryptographers
 - Crowds

Anonymous Communication



We want freedom of speech!

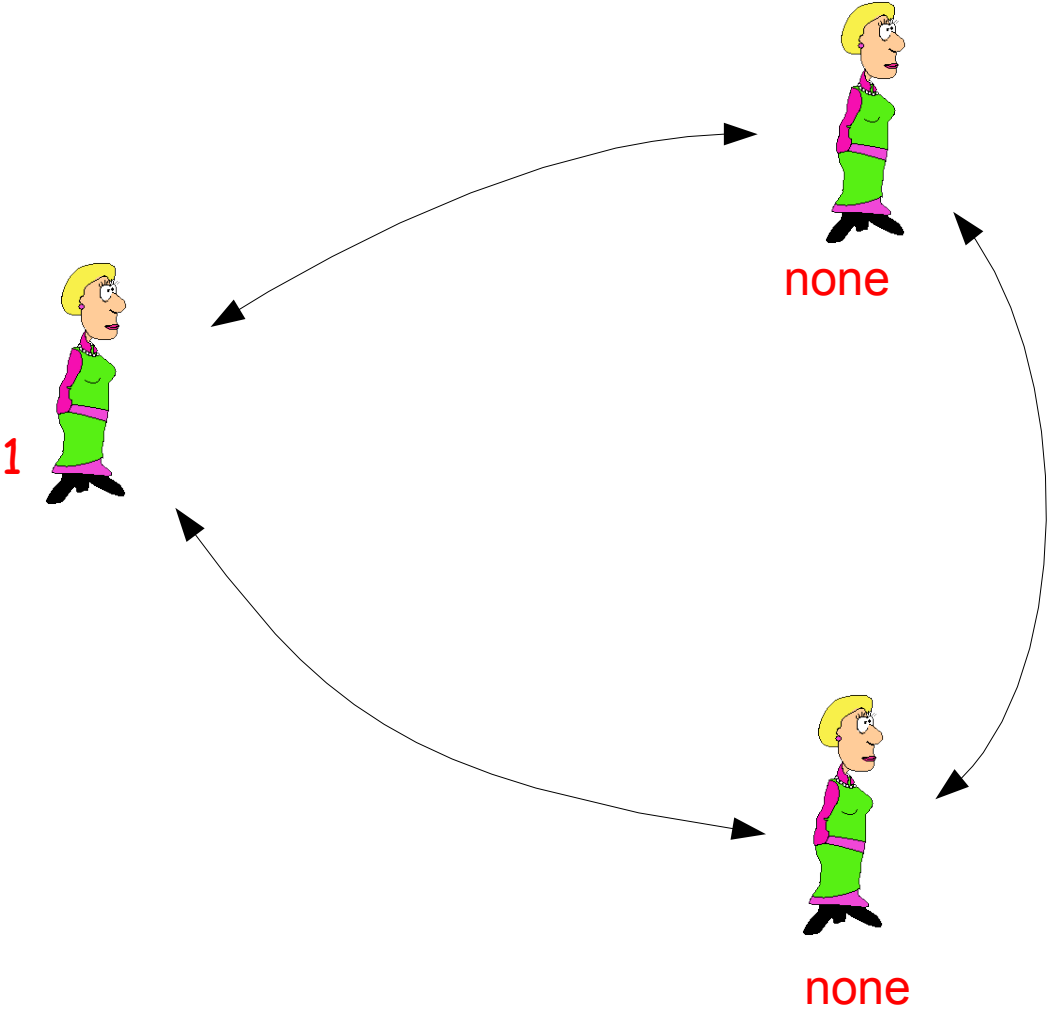
We want private voting!

However, communication can be linked by IP addresses and messages!

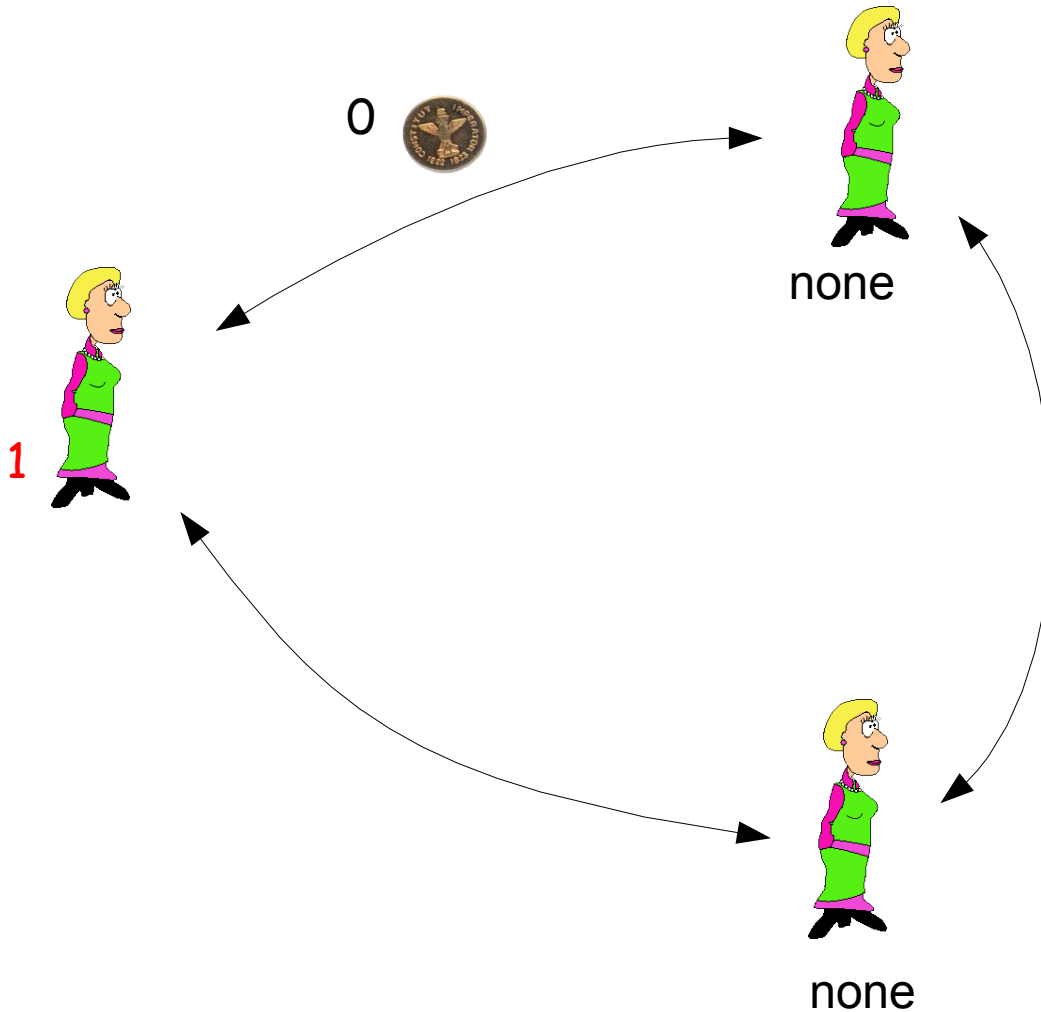
→ Encrypt and randomize messages

→ Network should hide IP address

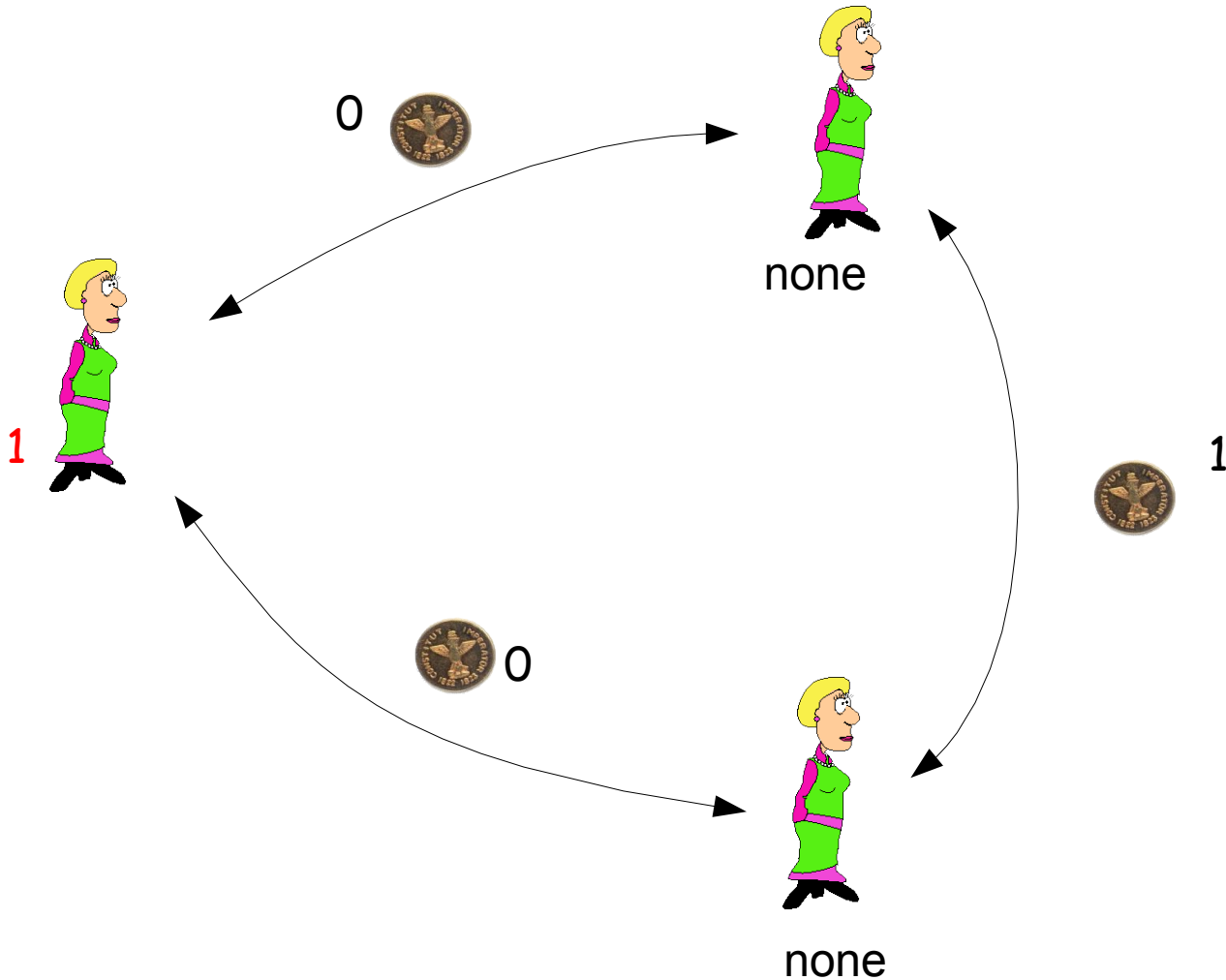
Dining Cryptographers: Inputs



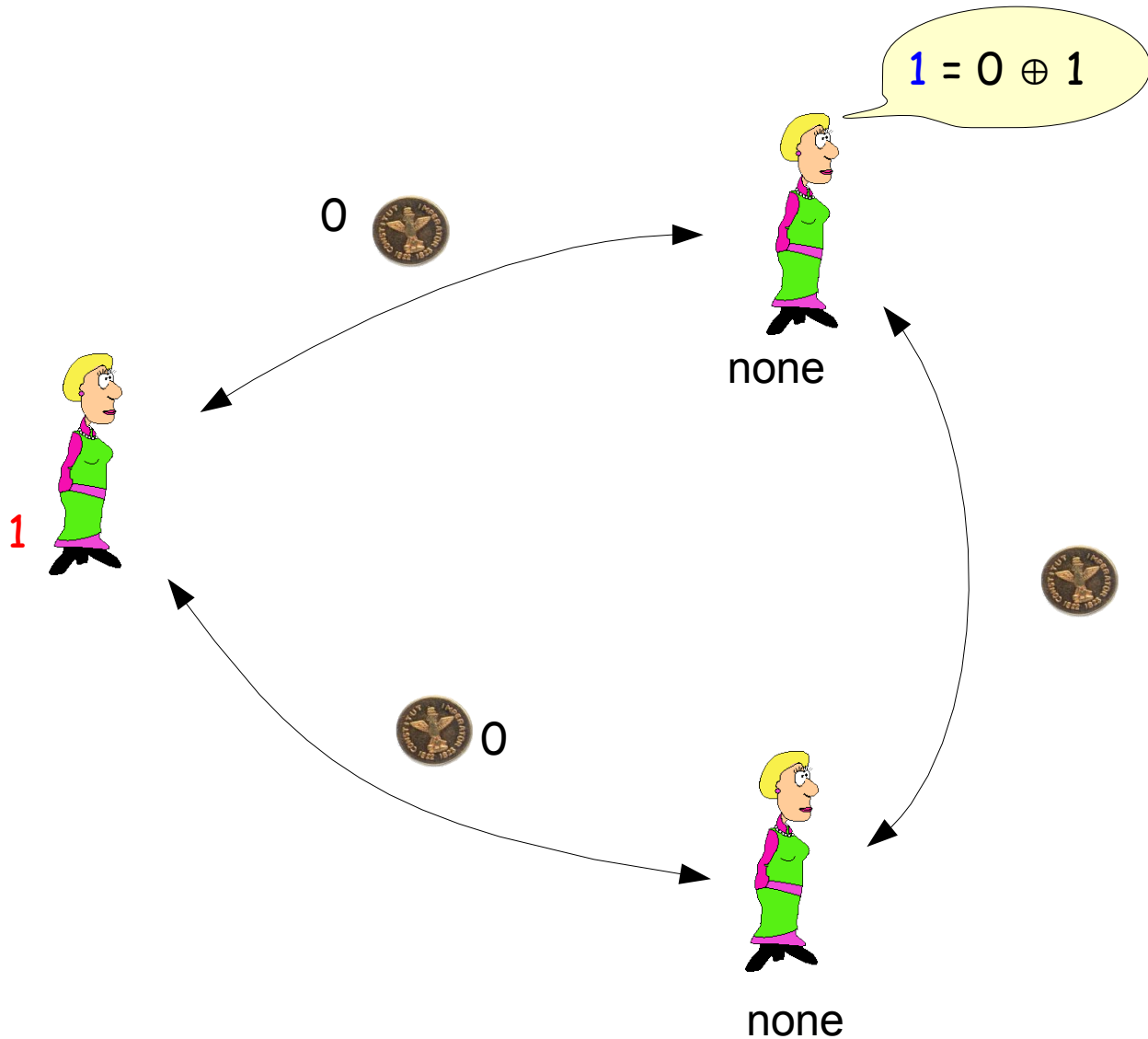
Dining Cryptographers: Flip Pairwise Coins



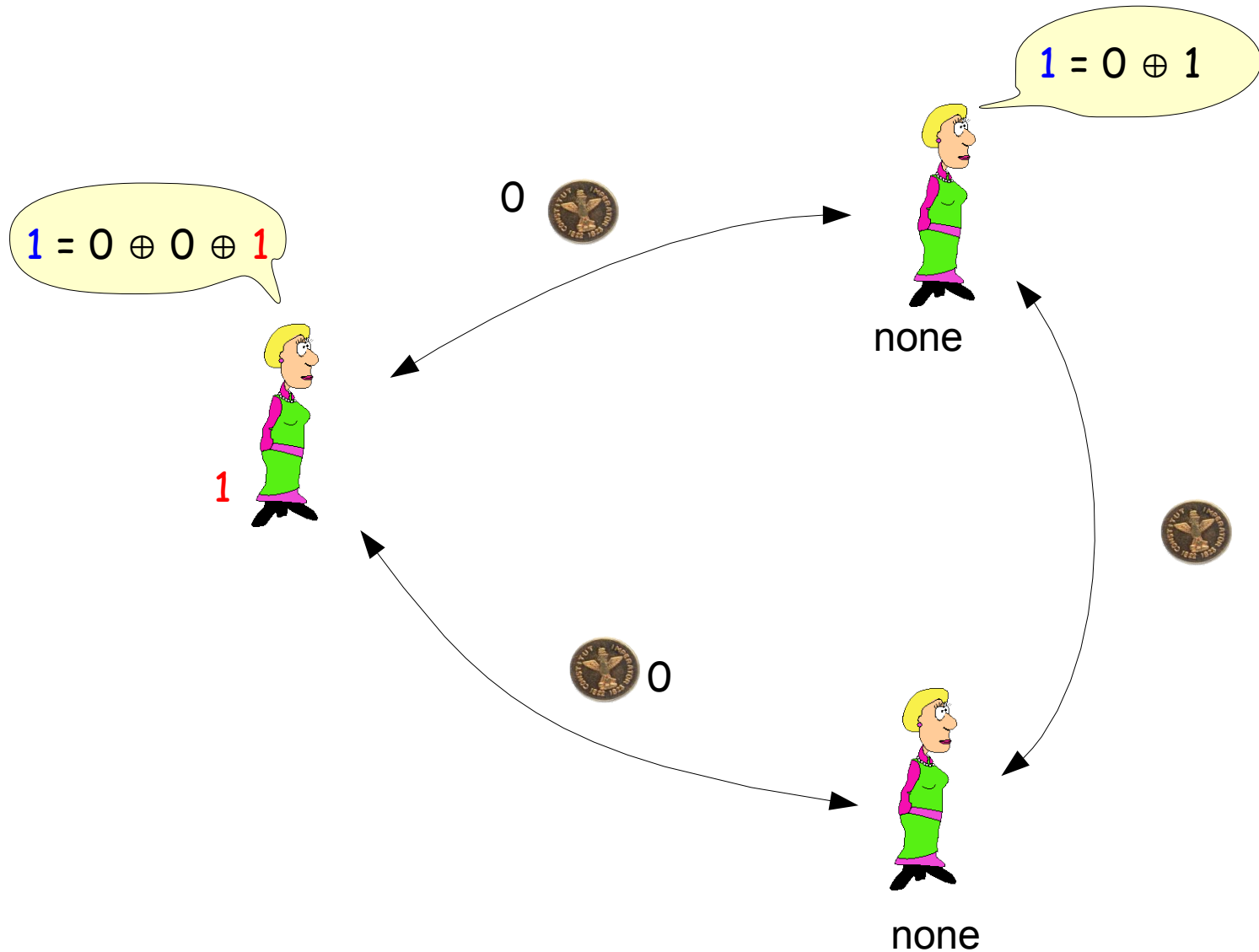
Dining Cryptographers: Flip Pairwise Coins



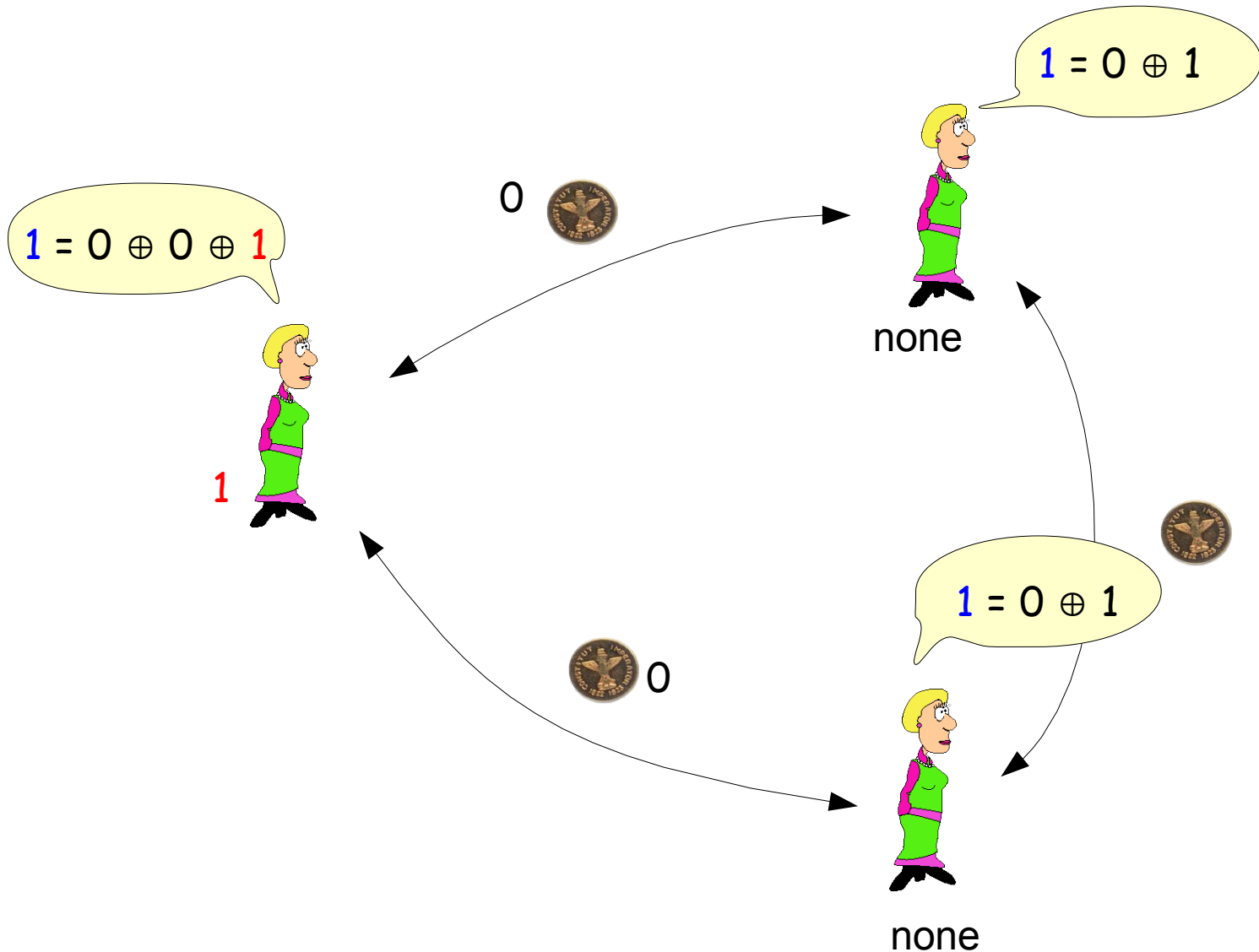
Dining Cryptographers: Announce Results



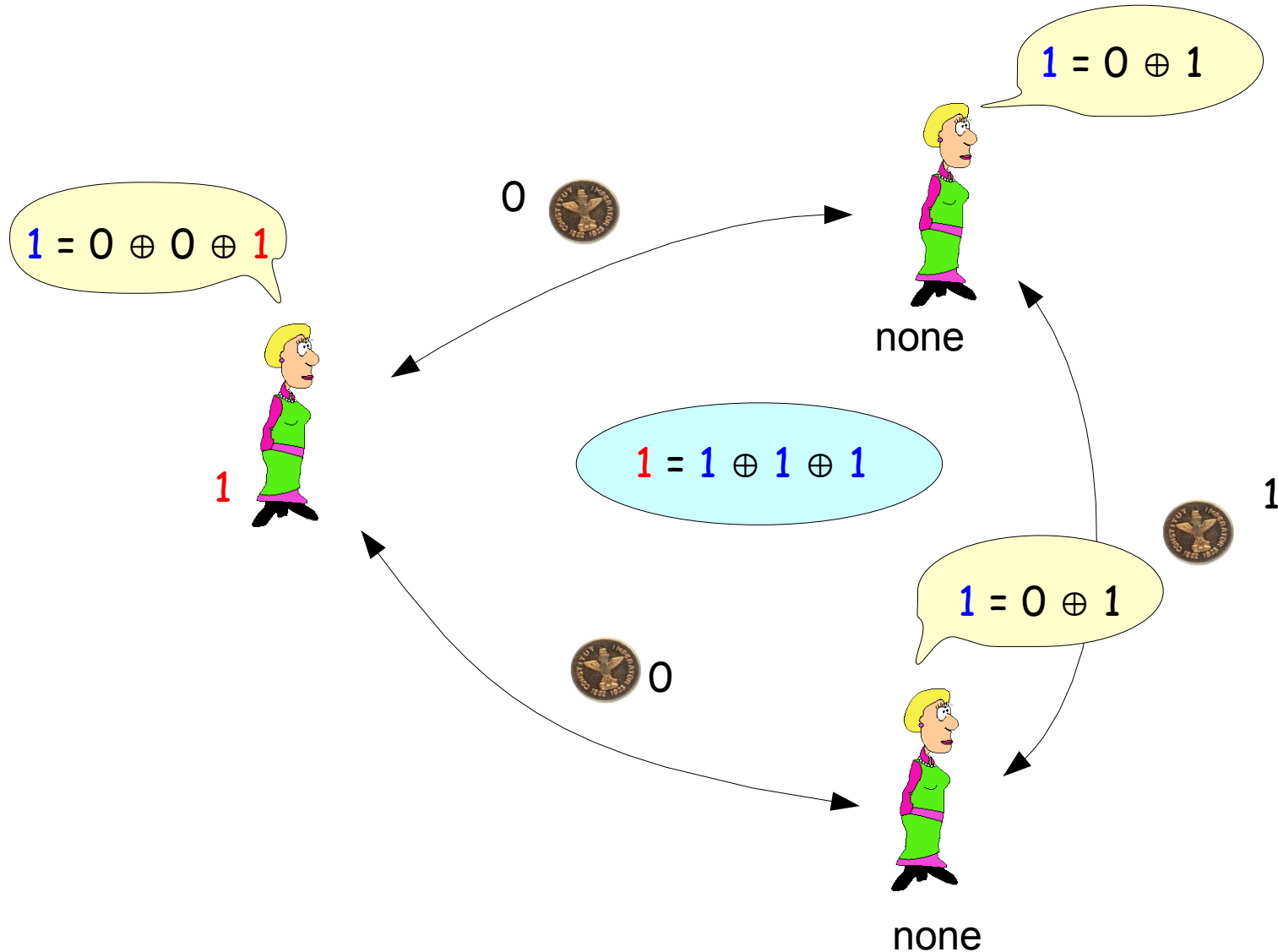
Dining Cryptographers: Announce Results



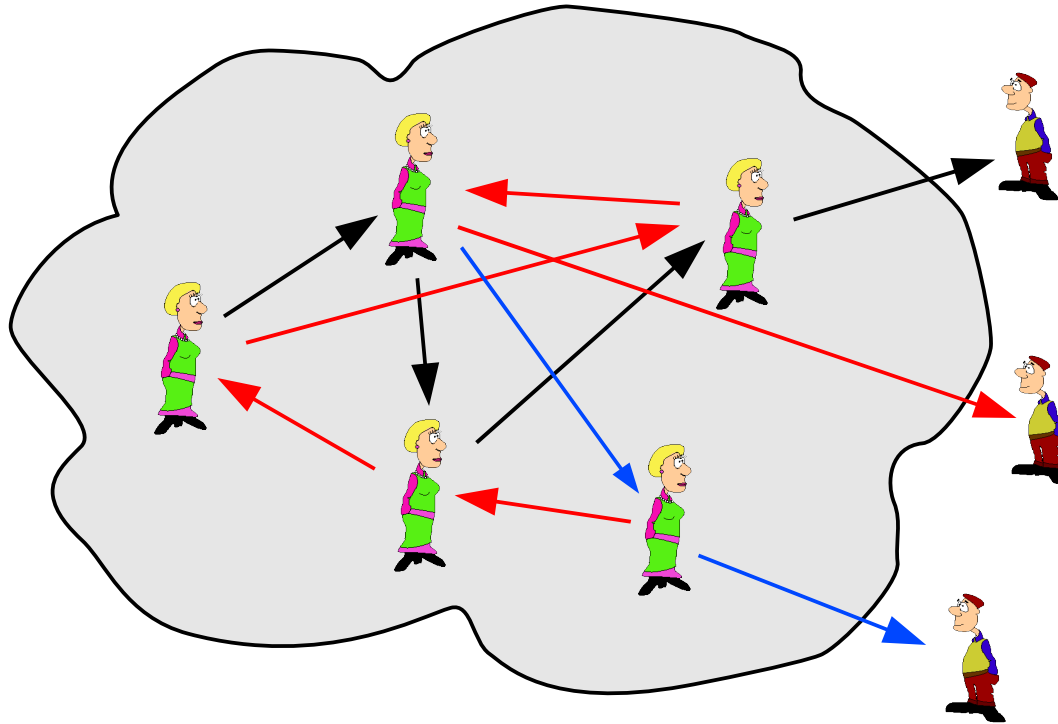
Dining Cryptographers: Announce Results



Dining Cryptographers: Compute Sent Msg

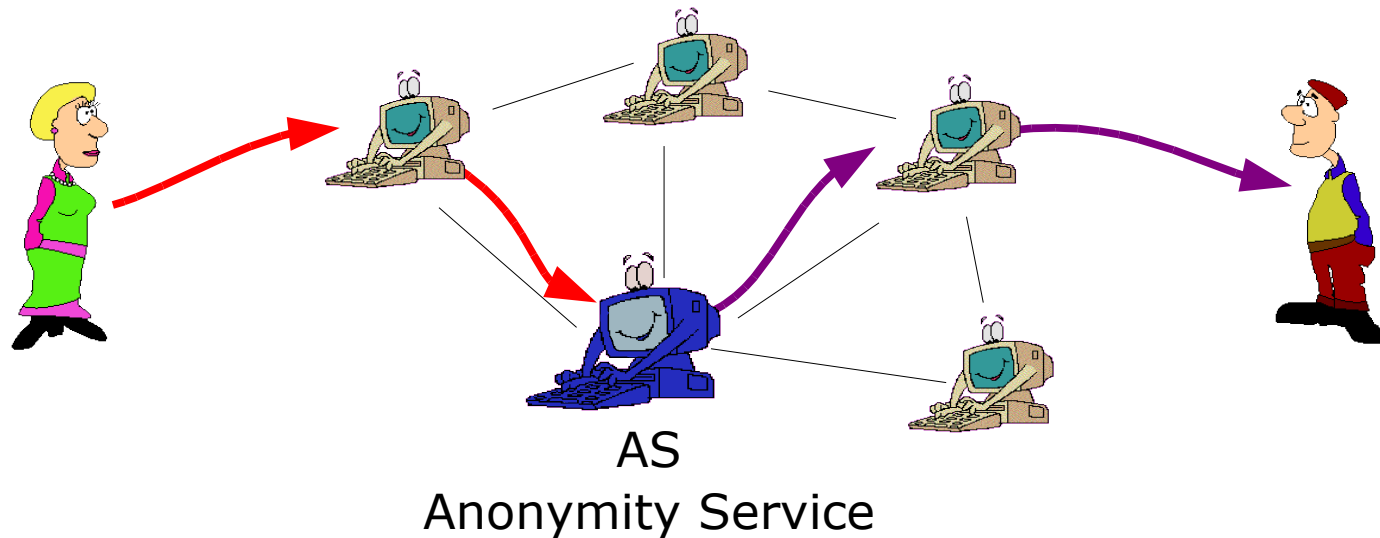


Crowds



- Each user sends message (http request) to other user
- When receiving message from another user, flip a coin. If it's tails send message to server (make http request). If heads, send message to another (random) user.
- Message travels back the same way it came
- Requires central (trusted) registration server

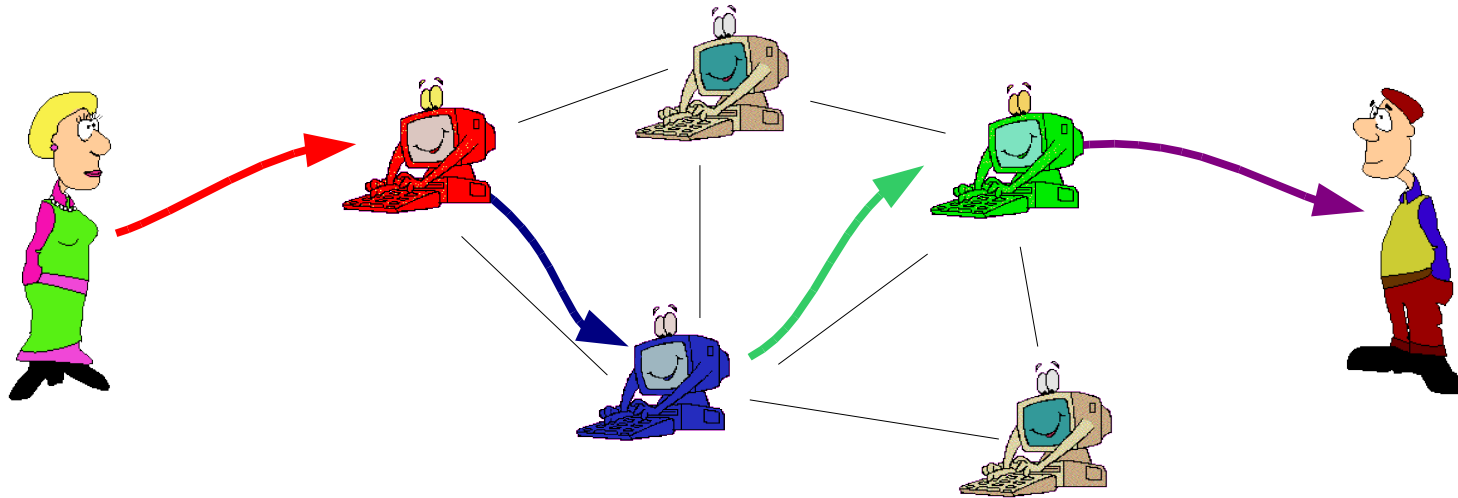
Anonymous Communication



Offer Anonymity Services

- Proxy and encryption (e.g., SSL)
 - Send Web-request via SSL to AS, AS gets and provides page
 - AS learns contents.
- Re-encryption
 - Alice sends $c_1 = \text{Enc}_{AS}(\text{Bob}, \text{Enc}_{\text{Bob}}(m))$ to AS
 - AS decrypts c_1 and forwards $\text{Enc}_{\text{Bob}}(m)$ to Bob.

Mix Networks – To Weaken Trust in AS

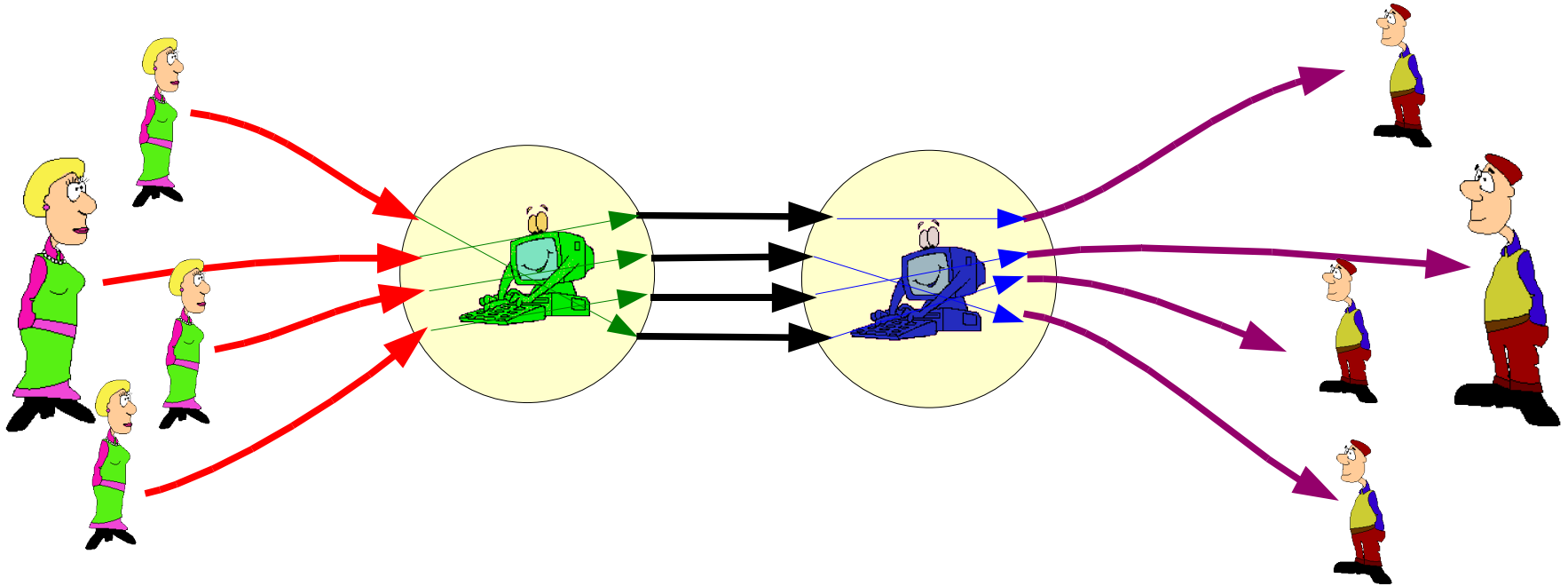


Route over several Anonymity Services: Need to trust only one

- Alice selects route she likes
- Alice sends to first router (**A1**)
$$c_1 = \text{Enc}_{A_1}(A_2, \text{Enc}_{A_2}(A_3, \text{Enc}_{A_3}(\text{Bob}, \text{Enc}_{\text{Bob}}(m))))$$
- Each AS peels off layer and forwards.
- Still, there needs to be more than one message at the same time

NOTE: SECURITY AGAINST CHOSEN CIPHERTEXT NEEDED!!!

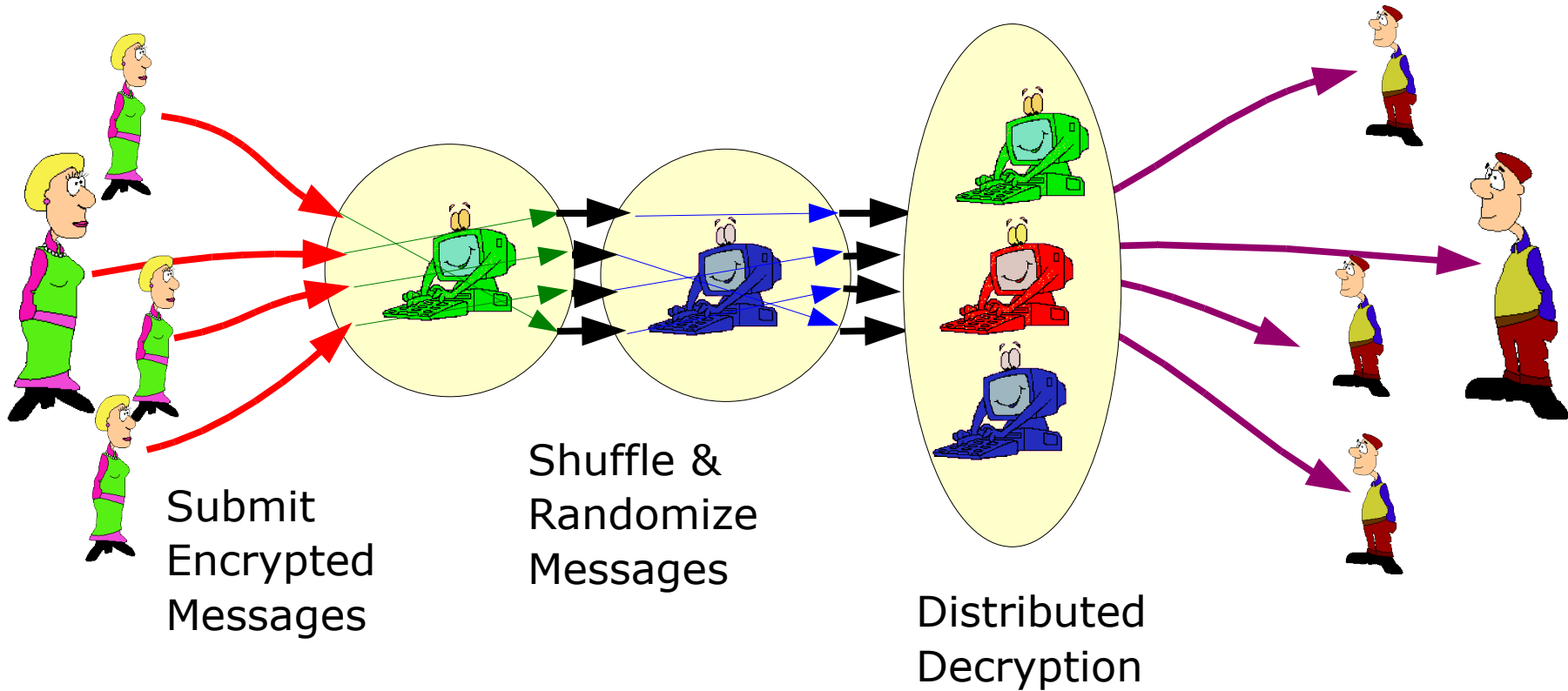
Mix Networks – Variations



Variations:

- Free path (onion routing) vs fixed path (mixes)
- Prove that you mixed correctly or not
- Fixed Batches vs Real-time mixing strategies

Provable Mixnets



Provable Mixnets

- Use 2-Key ElGamal
- Assumes availability of secure bulletin board
- Decryption server jointly generate public key
 - Server's public key $y_j = g^{x_j}$ and $z_j = g^{w_j}$,
 - joint public keys $y = \prod_j y_j$ and $z = \prod_j z_j$
- User i encrypts and post their message on *bulletin board*
 - choose random $r_1, r_2 \in \{1, \dots, q\}$;
 - compute $c^{(i)} = (y^{r_1} m, g^{r_1}, z^{r_2} m, g^{r_3}, \text{proof})$
 $= (c_1^{(1)}, \dots, c_5^{(1)})$
- List of ciphertexts $c^{(1)}, \dots, c^{(k)}$ as input the network

Provable Mixnets

- Verify proofs in ciphertexts
- Randomize and shuffle
 - randomize ciphertext: choose $r1 \in \{1, \dots, q\}$;
 - compute $c'(i) = (c1^{(i)} y1^{r1}, c2^{(i)} g^{r1})$
 $= (c'1^{(i)}, c'2^{(i)})$
 - chose random π permutation $d^{(\pi(i))} = c'(i)$
 - prove correctness of $d^{(j)}$ (in principle as follows):
PK $\{(\rho i, \pi i) :$
$$d1^{(\pi(i))} = c'1^{(i)} y1^{\rho i} \wedge d2^{(\pi(i))} = c'2^{(i)} g^{\rho i} \}$$

(proof of knowledge of permutation needs some extra tricks which are out of scope here (see, e.g., Neff01)).

Provable Mixnets

- Verifiable Decryption

- Given as inputs $d^{(i)} = (d1^{(i)}, d2^{(i)})$

- each decryption service provider computes

$$d2^{(i,j)} = d2^{(i)} \times_j$$

- and proves correctness

$$\text{PK}\{(\xi_j) : d2^{(i,j)} = d2^{(i)} \xi_j \wedge \gamma_i = g^{\xi_j}\}$$

- Combine and Decrypt

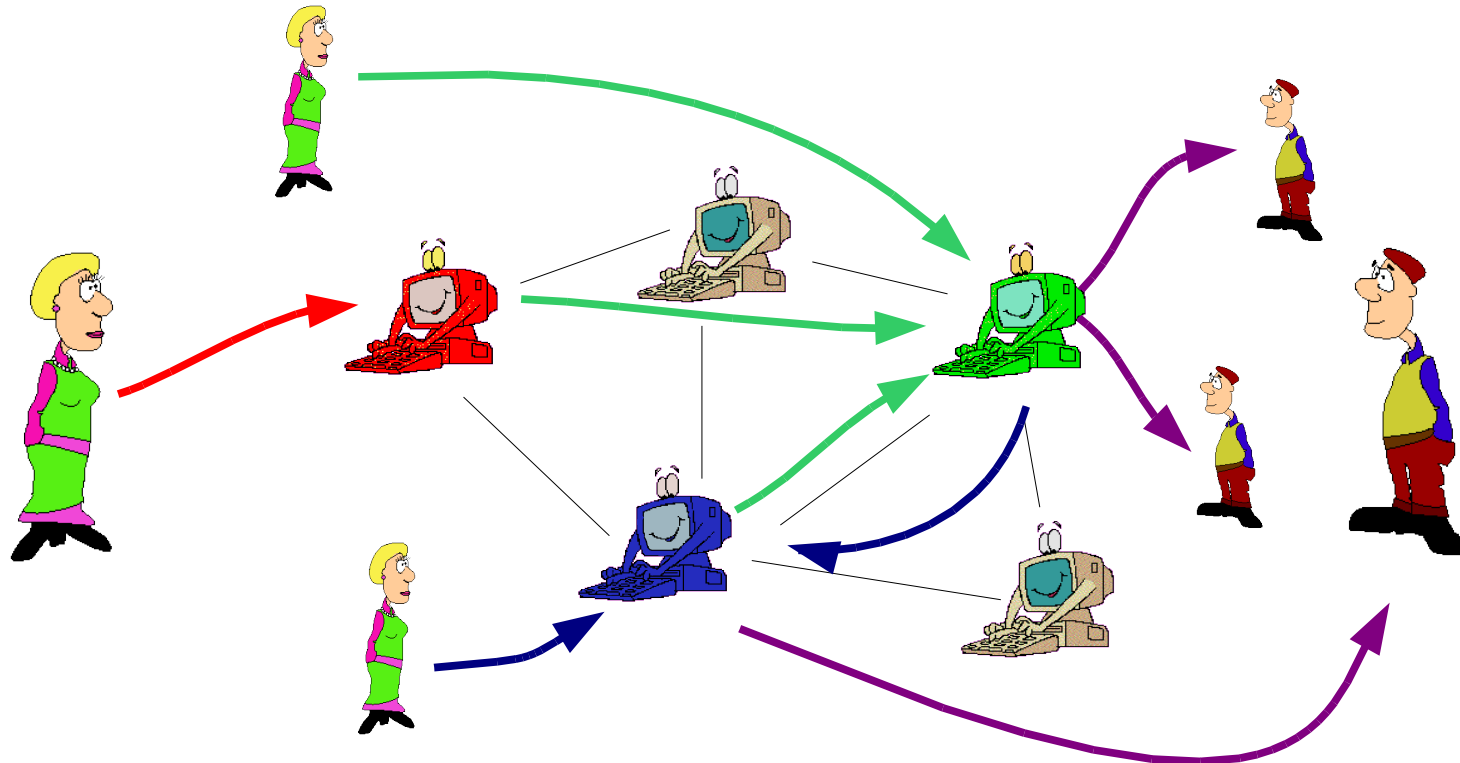
- $m^{(i)} = d1^{(i)} / \prod_j d2^{(i,j)}$

$$= d1^{(i)} / \prod_j d2^{(i)} \times_j = d1^{(i)} / \prod_j (g^{r1}) \times_j$$

$$= \gamma^{r1} m^{(i)} / (\prod_j g \times_j)^{r1} = \gamma^{r1} m^{(i)} / \gamma^{r1} = m^{(i)}$$

- (One can use so-called Threshold Encryption scheme to protect against non-cooperative routers.)

Onion Routing (Free Routes)



1. Each User chooses own route
2. Server peels of layer of encryption, discovers whom to send next
3. Problem:
 - How to mix
 - Encryption should not reveal number of (current) layers

Onion Routing

Problem: size of ciphertexts depends on # of remaining routers

$$\text{Enc}_{A_1}(A_2, \text{Enc}_{A_2}(A_3, \text{Enc}_{A_3}(\text{Bob}, \text{Enc}_{\text{Bob}}(m))))$$

- Need to add name of next recipient
- And typical encryption scheme expands length (cf. ElGamal where an encryption of one group element results in a ciphertext of two group elements)

Onion Routing

Solution:

- Notation:

- $\{m\}_k$ denotes the symmetric encryption of m under key k
- $\{e\}_{k^{-1}}$ denotes the symmetric decryption of e under key k

- Assume sender shares symmetric keys k_i with each router:

$$O1 = \{\{\{\{m\}_{k4}\}_{k3}\}_{k2}\}_{k1}, \{\{\{Bob\}_{k3}\}_{k2}\}_{k1}, \{\{A3\}_{k2}\}_{k1}, \{A2\}_{k1}$$

- Receiving $O1$, Router $A1$ will compute:

$$O'1 = \{\{\{m\}_{k4}\}_{k3}\}_{k2}, \{\{\{Bob\}_{k3}\}_{k2}\}_{k2}, \{A3\}_{k2}, A2$$

- $A1$ needs to grow onion again: chose random R and comp.

$$O2 = \{\{m\}_{k4}\}_{k3}\}_{k2}, R, \{\{\{Bob\}_{k3}\}_{k2}\}_{k2}, \{A3\}_{k2}$$

- $A1$ sends $O2$ to $A2$

Onion Routing

- Receiving

$$O_2 = \{\{\{m\}_{k_4}\}_{k_3}\}_{k_2}, R, \{\{Bob\}_{k_3}\}_{k_2}, \{A_3\}_{k_2}$$

A2 computes

$$O'_2 = \{\{m\}_{k_4}\}_{k_3}, \{R\}_{k_2-1}, \{Bob\}_{k_3}, A_3$$

- A2 needs to regrow onion: choose random R' and compute

$$O_3 = \{\{m\}_{k_4}\}_{k_3}, R', \{R\}_{k_2-1}, \{Bob\}_{k_3}$$

- And so on. At some point Bob will receive:

$$O_4 = \{m\}_{k_4}, R'', \{R'\}_{k_3-1}, \{\{R\}_{k_2-1}\}_{k_3-1}$$

He computes

$$O'_4 = m, \{R''\}_{k_4-1}, \{\{R'\}_{k_3-1}\}_{k_4-1}, \{\{\{R\}_{k_2-1}\}_{k_3-1}\}_{k_4-1}$$

and, as $\{\{\{R\}_{k_2-1}\}_{k_3-1}\}_{k_4-1}$ does not match any name, outputs m

Onion Routing

- How are keys distributed:

$$O1 = \{\{\{\{m\}_{k4}\}_{k3}\}_{k2}\}_{k1}, \{\{\{Bob\}_{k3}\}_{k2}\}_{k1}, \{\{A3\}_{k2}\}_{k1}, \{A2\}_{k1}$$

Replace, e.g., A3 by (A3,C3), where C3 is encryption of k3 under A3's *public key*

- What about chosen ciphertext security? I.e., ciphertext must not be changeable!
 - To make ciphertext non-malleable, include hash of onion as label in public key encryption
 - -> Routers *cannot* generate R at random
 - choose it pseudo random, e.g., $\{0\}_{k3-1}$

Onion Routing

- So Onion O4 is computed as follows:

$$O4 = \underbrace{\{m\}_{k4}, \{0\}_{k3-1}, \{\{0\}_{k2-1}\}_{k3-1}, \{\{\{0\}_{k1-1}\}_{k2-1}\}_{k3-1}}_{L4}, (C4, Bob),$$

where $C4 = Enc_{PK_{Bob}}(k4, Hash(L4))$

- So Onion O3 is computed as follows:

$$\begin{aligned} O3 &= \{\{m\}_{k4}\}_{k3}, \{\{\{0\}_{k2-1}\}_{k3-1}\}_{k3}, \{\{\{\{0\}_{k1-1}\}_{k2-1}\}_{k3-1}\}_{k3}, \\ &\quad \{C4, Bob\}_{k3}, (C3, A3) \\ &= \{\{m\}_{k4}\}_{k3}, \{0\}_{k2-1}, \{\{0\}_{k1-1}\}_{k2-1}, \{C4, Bob\}_{k3}, (C3, A3), \end{aligned}$$

where $C3 = Enc_{PK3}(k3, Hash(L3))$

Anonymity Services

- Browser Proxies
 - <http://www.alphasquad.de/anony.html>
 - Many others
- Re-Encryption
 - JAP <http://anon.inf.tu-dresden.de/>
 - Tor <http://tor.eff.org/>
- Provable Re-Encryption
 - Robust Mixnets (numerous proposals...)
- Other
 - Crowds

References

- D. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*. In Communications of the ACM.
- D. Chaum: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*. Journal of Cryptology, 1988.
- J. Camenisch and V. Shoup: *Practical Verifiable Encryption and Decryption of Discrete Logarithms*. In Advances in Cryptology - CRYPTO 2003.
- J. Camenisch and A. Lysyanskaya: *A Formal Treatment of Onion Routing*. In Advances in Cryptology - CRYPTO 2005.
- J. Camenisch and A. Mityagin: *Mix-Network with Stronger Security*. In Privacy Enhancing Technologies – PET 2005.
- T. ElGamal: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. In Advances in Cryptology - CRYPTO '84.